멀티미디어 저장 시스템에서 참조 유형을 고려한 혼성 버퍼 교체 기법

류 연 승[†]

요 약

멀티미디어 저장 시스템을 위한 기존의 버퍼 캐시 기법들은 멀티미디어 파일의 순차적 접근만을 고려하고 반복 참조는 고려하지 않고 있다. 그러나, 외국어 영상 학습의 경우 사용자가 어떤 장면을 반복 구간으로 설정하면 자동으로 수회 반복 상영하는 기능이 있을 수 있다. 본 논문에서는 순차 참조와 반복 참조가 혼재하는 멀티미디어 저장 시스템을 위한 새로운 버퍼 교체 기법인 HBM(Hybrid Buffer Management)을 제안한다. 제안한 기법은 응용 수준에서 파일의 참조 유형을 파일 시스템에게 알려주고 HBM이 각 파일에 적절한 버퍼 교체 정책을 적용한다. 실험 결과, HBM은 DISTANCE나 LRU와 같은 이전 버퍼 교체 기법보다 캐시 적중률을 높일 수 있음을 알 수 있었다.

Hybrid Buffer Replacement Scheme Considering Reference Pattern in Multimedia Storage Systems

Ryu Yeon Seung[†]

ABSTRACT

Previous buffer cache schemes for multimedia storage systems only exploited the sequential references of multimedia files and didn't consider looping references. However, in some video applications like foreign language learning, users mark the scene as loop area and then application automatically playbacks the scene several times. In this paper, we propose a new buffer replacement scheme, called HBM(Hybrid Buffer Management), for multimedia storage systems that have both sequential and looping references. Proposed scheme assumes that application layer informs reference pattern of files to file system. Then HBM applies an appropriate replacement policy to each file. Our simulation experiments show that HBM outperforms previous buffer cache schemes such as DISTANCE and LRU.

Key words: Multimedia Storage Systems, Reference Pattern, Buffer Cache, Cache Hit Ratio

1. 서 론

컴퓨터와 통신 기술의 발전으로 인해 초고속 인터넷을 통하여 멀티미디어 데이터를 실시간으로 전송하는 응용이 실현 가능하게 되었다. 멀티미디어 데이터는 그 크기가 매우 크고, 비교적 오랜 시간동안 연속적으로 처리되어야 하는 특징을 가지고 있다. 이러한 데이터를 저장하는 멀티미디어 저장 시스템은 고객들의 실시간 요구 조건을 만족시키면서 서비스하는 동시 고객 수를 최대화해야 한다[1-3].

일반적으로 저장 시스템은 디스크 입출력의 성능 향상을 위해 버퍼 캐시를 사용하고 있다. 버퍼 캐시는 디스크로부터 읽은 데이터를 임시로 저장하고 미래의 요청에 대응함으로써 시스템의 성능을 향상시킬 수 있다. 보통 디스크의 접근 시간보다 메모리의 접근 시간이 더 빠르기 때문에 데이터를 디스크 대신에 메모리에서 읽을 수 있다면 데이터에 대한 응답시

이 논문은 2001년도 한림대학교 교비연구비에 의하여 연구되었음

† 정회원, 한림대학교 정보통신공학부 전임강사

간을 빠르게 할 수 있다. 그러나, 버퍼 공간이 제한되어 있으므로 버퍼 교체 작업이 필요하다. 다양한 환경에서 버퍼 캐시 적중률(hit ratio)을 높이기 위한 많은 버퍼 교체 기법들이 연구되어 왔다[4-11].

멀티미디어 파일 시스템을 위한 여러 버퍼 교체 기법들도 제안되었다[12-14]. 이러한 버퍼 교체 기법들은 대부분 데이터의 순차적인 접근에 착안한 기법들이다. 예를 들어, (그림 1)과 같이 두 고객이 멀티미디어 파일 1을 참조하고 있다고 가정하자. 각 고객은 파일을 참조하게 되면 파일의 처음부터 끝까지 순차적으로 참조한다. 먼저 고객 1이 파일을 참조하여 현재 블록 11을 참조하고 있으며, 고객 2가 뒤이어 파일을 참조하여 블록 8을 참조하고 있다. 이와 같이 멀티미디어 파일의 순차 참조에서는 각 고객은 한번 참조한 데이터를 다시 참조하지 않으며, 대신에 늦게 같은 파일을 요청한 다른 고객이 참조하는 특징을 보인다. 제안된 기법들은 선행 고객이 참조한 데이터를 버퍼에 캐시하고 후행 고객이 참조하게 하는 방법을 사용하여 버퍼 적중률(buffer hit ratio)을 높였다. 그림에서 고객 1이 참조하는 블록들 중에서 최근에 참조한 블록 3개를 버퍼에 캐시한다면 고객 2는 필요한 모든 블록을 버퍼에서 참조하게 된다. 이와 같은 고객 간의 순차적인 블록 구간을 캐시하는 기법은 멀티미디어 파일의 순차적인 참조에서 최적에 가까운 성능을 나타내는 것으로 알려져 있다[12].

본 연구에서는 순차 참조 유형의 파일과 반복 참조 유형의 파일이 함께 저장되어 있는 멀티미디어 저장 시스템에서 버퍼 캐시 성능을 향상시키는 혼성 버퍼 교체 기법(Hybrid Buffer Management : HBM)을 연구하였다. 최근 다양한 멀티미디어 응용들이 생겨나면서 멀티미디어 파일을 순차적으로 참조하는 경우 뿐만 아니라 반복적으로 참조하는 응용들도 출현하고 있다. 예를 들면, 동영상을 사용한 원격 교육 시스템을 들 수 있다. 특히, 외국어 교육의 경우, 교육생이 대화 장면을 반복 학습하기 위해서 대화 장면을 반복 구간으로 지정하면 응용 프로그램이 반복 구간을 자동으로 일정 횟수를 반복하는 예가 있다. 이 경

그림 1. 멀티미디어 파일에서 데이터 블록의 참조 예

우, 파일의 블록들을 반복적으로 참조하게 된다.

그러나, 기존의 연구들은 순차 참조의 경우만 고려했고 반복 참조가 있는 경우에 대한 연구는 거의 없다. 반복 참조가 있다면 기존의 순차 참조만을 고려한 버퍼 관리 기법은 적합하지 않게 되며 버퍼 적중률을 높일 수 있는 새로운 기법이 필요하다. 본 연구에서는 참조 유형별로 적절한 버퍼 교체 정책을 사용하여 버퍼 적중률을 향상시키는 방법을 제안하였다.

제안한 기법은 응용 수준에서 파일의 참조 유형을 파일 시스템에 알려주며, 파일 시스템이 파일별로 적합한 버퍼 교체 기법을 적용하는 방법이다. 본 연구에서는 실험을 통해 순차 참조와 반복 참조가 있을 때 기존 버퍼 교체 기법의 성능을 살펴보고 새로운 버퍼 교체 기법의 필요성을 보였다. 또한, 제안한 혼성 버퍼 교체 기법이 기존 기법보다 좋은 버퍼 성능을 나타냄을 보였다.

본 논문의 구성은 다음과 같다. 2장에서 관련 연구를 살펴본다. 3장에서는 새로운 버퍼 관리 기법인 HBM을 설명한다. 4장에서는 기존 버퍼 교체 기법과 HBM 기법의 실험 결과를 설명한다. 마지막으로, 결론과 향후 연구를 5장에서 기술한다.

2. 관련 연구

멀티미디어 파일 시스템에서 버퍼에 관한 많은 연구가 있어 왔다. 이 연구에는 디스크 스케줄링과 연관되어 요구 버퍼를 계산하고 예약하는 연구[14-19], 버퍼 사용률을 높이거나 요구 버퍼 크기를 줄이는 연구[20,21], 데이터 선반입(prefetch) 연구[22], 버퍼 캐시의 교체 기법에 관한 연구[12-14] 등이 있다. 본 논문에서는 버퍼 캐시의 교체 기법에 관한 연구를 다루며 여기서는 이와 관련한 기존 연구를 살펴본다.

버퍼 캐시의 성능은 데이터 참조 유형과 버퍼 교체 정책에 달려있다.

데이터 참조 유형에는 크게 순차(sequential) 참조, 반복(looping) 참조, 시간 국부성(temporal localized) 참조, 확률적(probabilistic) 참조가 있다. 순차 참조는 모든 데이터가 순차적으로 참조되며 한번 참조한 데이터가 다시 참조되지 않는 참조 유형을 말한다. 반복 참조는 정기적인 간격으로 반복해서 데이터가 참조되는 참조 유형을 말한다. 시간 국부성 참조는

최근에 참조되는 블록이 가까운 미래에 참조되는 참조 유형을 말한다. 시간 국부성 참조의 경우 Least Recently Used (LRU) 정책이 최적(optimal)에 가까운 버퍼 교체 정책으로 알려져 있다[4]. 확률적 참조는 독립 참조 모델(Independence Reference Model)에 의해 데이터가 참조되는 참조 유형이다. 이때, 각 블록 i는 확률 p_i를 가지고 확률에 의해 독립적으로 참조된다.

최적의 버퍼 교체 기법은 버퍼 용량이 정해져있을 때 캐시 적중률을 최대화할 수 있도록 데이터를 교체하는 기법이다. 최적의 교체 기법은 가장 오랜 시간동안 참조하지 않을 데이터를 교체하는 방법이다. 그러나, 미래의 참조에 대한 지식이 필요하기 때문에 현실적으로 구현이 어렵다. 따라서, 최적의 교체 기법에 근사하는 LRU, Most Recently Used (MRU), Least Frequently Used (LFU) 등에 기반한 여러 기법들이 제안되었다[8-11].

멀티미디어 파일 시스템을 위한 대표적인 버퍼 교체 기법으로는 BASIC/DISTANCE 기법[12]과 인터벌 캐싱(Interval Caching) 기법[13,14] 등이 제안되었다. 제안한 기법들은 멀티미디어 파일의 순차적 참조 유형에 최적에 가까운 좋은 성능을 보이며, LRU 정책을 사용하면 매우 좋지 않은 성능(90% 이상의 캐시 부재율)을 나타냄을 보였다[12].

Özden 등이 제안한 BASIC 기법은 최적의 버퍼 교체 기법과 비슷하며, 현재 서비스 중인 고객에 의해 가장 오랜 시간동안 참조되지 않을 블록을 가진 버퍼를 교체하는 기법이다. 이 기법은 고객마다 상태 정보를 유지하고 매 서비스 사이클마다 고객의 상태 정보를 이용하여 모든 버퍼에 대하여 다음에 참조될 시간을 계산한다. 이 방법은 최적의 성능을 보여주지만, 버퍼마다 미래의 참조 시간 값을 서비스 사이클마다 계산해야 하므로 실행 오버헤드가 큰 문제가 있다. DISTANCE 기법은 선행 고객이 참조한 데이터 블록들을 버퍼 캐시에 저장하여 후행 고객이 버퍼에서 데이터를 참조할 수 있게 한 기법이다. 이때, 선행 고객과 후행 고객간의 데이터 블록 개수를 distance라고 정의하였다. 이 기법은 버퍼를 교체할 때 distance가 가장 큰 고객의 버퍼를 선택한다. 이 방법은 distance가 작은 블록을 버퍼에 유지시킴으로서 많은 고객이 버퍼에서 데이터를 참조할 수 있게 해준다.

Dan 등은 비디오 파일 서버를 위한 캐시 방법으로 인터벌 캐싱 기법을 제안하였다. 인터벌 캐싱도 선행 고객이 읽은 데이터를 버퍼에 유지시켜 후행 고객이 사용하게 하는 기법이다. 인터벌이란 같은 비디오 객체를 참조하면서 연속된 두 고객 간의 데이터 블록으로 정의된다. 연속적인 두 고객 간의 인터벌이 캐시되었다면 후행 고객은 선행 고객이 읽은 데이터를 버퍼에서 읽게 된다. 인터벌 캐싱 기법도 인터벌이 작은 고객부터 캐시하여 한정된 버퍼 공간에 보다 많은 수의 연속 고객을 유지시킴으로서 캐시 적중률을 최대화한다.

DISTANCE 기법과 인터벌 캐싱 기법은 다음과 같은 점에서 매우 유사하다. 첫째, 멀티미디어 파일의 참조는 순차적이라고 가정한다. 둘째, 같은 멀티미디어 파일을 참조하는 연속적인 두 고객 간의 데이터 블록을 캐시한다. 세째, 두 고객간의 인터벌(또는 distance)이 작은 것의 블록을 버퍼에 유지시킴으로서 버퍼 적중률을 높인다.

최근에는 전통적인 파일 시스템 분야에서 서로 다른 참조 유형을 보이는 응용마다 적절한 버퍼 교체 정책을 적용하는 기법들이 연구되었다[23-25]. 이러한 기법들은 참조 유형의 정보를 필요로 하는데, 참조 유형을 알아내기 위한 방법으로는 사용자 수준에서 정보를 제공하는 방법[23]과 수행 중에 참조 유형을 발견하는 방법이 제안되었다[24]. Cao 등은 응용 프로그램이 자신의 버퍼 교체 정책을 시스템에 지시하는 인터페이스를 제안하였다[23]. 응용 프로그램은 LRU 또는 MRU 정책을 지정할 수 있다. Choi 등은 응용 프로그램의 참조 유형을 수행 중에 탐지하고 응용 프로그램별로 적절한 교체 정책을 적용하는 방법을 제안하였다[24].

3. HBM (Hybrid Buffer Management) 기법

3.1 시스템 모델

파일이 N개의 블록으로 구성되어 있으며 블록 1부터 블록 N까지 블록 번호가 매겨 있다고 하자. 고객이 파일을 참조하는 블록 번호들의 나열을 참조 열(reference string)이라 한다. 어떤 고객이 파일을 참조하면서 만드는 참조 열이 {1, 2, [illegible], 5, 6, [illegible], 9, 10, ...}이라 하자. 이 참조 열에서 블록 {3, 4}는 2회 반복 참조되었고, 블록 {7, 8}은 3회 반복 참조되었다. 반복 참조된 블록 구간을 루프

라고 한다. 루프의 블록 개수를 루프 길이라고 하고 루프를 반복 참조한 횟수를 루프의 횟수라고 한다. 루프 {3, 4}는 루프 길이가 2이고 루프 횟수는 2이다. 루프와 루프 사이에 순차 구간이 있으면 이 순차 구간의 블록 개수를 루프간 간격이라고 한다. 참조 열의 예에서 두 루프간 간격은 2이다. (루프 길이, 루프 횟수, 루프간 간격)을 루프 파라미터라고 부르겠다.

(그림 2)는 멀티미디어 파일에 대한 반복 참조의 예를 보이고 있다. 그림에서 고객 1은 블록 {3, 4}를 2회 반복 참조한 후 블록 {5, 6}을 순차 참조하다가 블록 {7, 8}을 3회 반복 참조하고 있다. 고객 2는 고객 1이 네 번째 블록을 참조할 때 파일을 참조하기 시작하여 블록 {3, 4}를 3회 반복 참조한 후 블록 5부터 다시 순차 참조하고 있다.

이와같이 멀티미디어 파일의 반복 참조에는 순차 참조와 반복 참조가 혼재하게 된다. 즉, 기본적으로는 순차 참조이면서 순차적으로 참조하는 도중에 특정 블록 구간을 여러번 반복 참조한다. 이와같은 참조 유형으로 많은 고객들이 파일을 접근하게 되면, 최근에 참조된 블록이 가까운 미래에 참조되는 시간 국부성 참조 유형을 보이게 된다. 시간 국부성 참조 유형에는 LRU가 최적에 가까운 버퍼 교체 정책으로 알려져 있으며[4], 연속하는 두 고객간의 인터벌(또는 distance)을 캐시하는 정책은 적합하지 않게 된다.

본 연구에서는 순차 참조 응용과 반복 참조 응용이 혼재하는 멀티미디어 저장 시스템을 가정한다. 순차 참조 응용은 파일을 첫 번째 블록부터 순차적으로 참조하는 응용이다. 반복 참조 응용은 기본적으로는 순차 참조 응용과 같으나 파일을 순차적으로 참조하는 도중에 반복 구간을 설정하고 설정한 구간을 여러 번 반복 참조하는 응용이다. 반복 참조 응용에서는 고객이 반복 구간을 설정하는 동안에도 서버는 다음 블록을 계속 전송하고 고객은 블록들을 상영할 것이다. 일단 반복 구간이 설정되면 고객으로부터 서버로

그림 2. 멀티미디어 파일에서 반복 참조의 예

메시지 통신이 있을 수 있다. 그러나, 이 메시지 통신으로 인한 오버헤드는 작기 때문에 무시할 수 있다. 메시지를 받은 서버는 반복 구간의 블록을 지정된 반복 횟수만큼 전송해야 한다.

파일의 속성에 참조 유형 속성이 있다고 가정한다. 또한, 파일의 참조 유형 속성을 응용 수준(application level)에서 설정할 수 있는 시스템 호출(system call)이 있다고 가정한다. 파일의 참조 유형은 그 파일을 사용하는 응용에 의해 설정된다. 즉, 파일을 사용하는 응용이 순차 참조 응용이라면 그 파일의 참조 유형을 순차 참조로 설정하고, 파일을 사용하는 응용이 반복 참조 응용이라면 그 파일의 참조 유형을 반복 참조로 설정한다.

3.2 버퍼 관리 방법

고객 i는 상영률 r_i로 데이터를 요청한다고 하자. 본 논문에서는 한 서비스 사이클 T 동안 각 고객들이 요청하는 $r_i * T$ 데이터를 디스크에서 읽어 버퍼에 저장하면 다음 사이클에서 각 고객에게 전송된다고 가정한다[1,2]. 이를 위해서 각 고객마다 최소한 이중 버퍼(double buffer)가 할당되어야 한다. 시스템의 전체 버퍼 공간은 n_B개의 버퍼로 구성된다. 각 버퍼의 크기는 d로서 같은 크기이다. 디스크에서 읽어오는 블록의 크기는 버퍼의 크기와 같다고 가정한다. 저장 시스템은 서비스 사이클마다 서비스 중인 각 고객들이 이전 사이클에서 사용한 버퍼를 가용 버퍼로 반환하고 다음 사이클에 필요한 데이터 블록을 버퍼에 읽어온다.

제안한 HBM은 파일의 참조 유형 속성을 이용하여 파일 별로 버퍼 교체 정책을 적용한다. 고객이 파일을 요청하여 참조하면 참조된 블록들이 버퍼에 저장되게 된다. 블록을 저장하고 있는 버퍼가 교체될 필요가 발생하면 파일의 버퍼 교체 정책에 의해 교체될 버퍼를 선택한다.

(그림 3)은 제안한 버퍼 관리 기법인 HBM의 구조를 보이고 있다. HBM은 시스템 버퍼 관리자, DISTANCE 관리자, LRU 관리자로 구성되어 있다.

시스템 버퍼 관리자는 새 고객의 버퍼 자원에 대한 수용 제어 및 수용되어 서비스되는 고객들을 위한 버퍼 관리를 수행한다.

시스템 버퍼 관리자는 버퍼 자원에 대한 수용 제어(admission control)를 수행한다. 즉, 시스템에 새
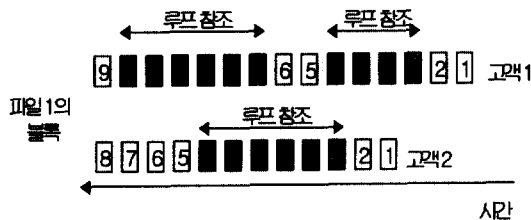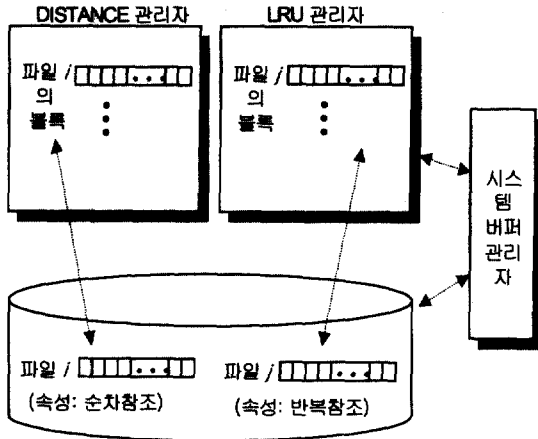
그림 3. HBM의 구조

고객이 도착하면 새 고객을 포함한 모든 고객에게 버퍼를 할당할 수 있는지를 검사한다. 새로 도착하는 고객을 포함하여 총 고객의 수를 M이라 하자. 이때, 모든 고객들이 요청하는 데이터를 버퍼에 저장할 수 있어야 하고 이중 버퍼를 사용하므로, 일반적으로

$$2T\sum_{i=1}^{M} r_i \le n_B \cdot d \quad (1)$$

가 만족되어야 한다. 만일, 식 (1)에서 등식이 성립되고 각 고객의 도착 간격이 $2T$ 보다 크다면 시스템의 모든 버퍼가 각 고객의 이중 버퍼로 사용되어야 하고 고객들이 공유할 수 없다. 따라서, 모든 고객은 서비스 사이클마다 데이터를 디스크에서 읽어야 하므로 캐시 적중률(hit ratio)이 0이 된다. 그러나, 식 (1)에서 오른쪽 항이 왼쪽 항보다 훨씬 크다면 가용 버퍼가 있게 되고 참조했던 블록을 임시로 저장할 수 있으므로 캐시로부터 데이터를 서비스할 수 있게 된다.

식 (1)로부터, M 고객들을 서비스하기 위해서 다음과 같은 버퍼가 있다면 새 고객을 수용할 수 있다.

$$\lceil 2\frac{T}{d}\sum_{i=1}^{M} r_i \rceil \quad (2)$$

수용된 고객은 파일을 참조할 수 있다. 참조되는 파일의 블록은 파일의 참조 유형 속성에 따라 다른 교체 정책으로 관리된다.

시스템 버퍼 관리자는 전체 버퍼를 각 관리자에게 할당해야 하는데, 관리될 파일의 고객 수에 비례하여 버퍼를 할당한다. 파일의 고객 수는 통계값을 사용하여 알 수 있다.

저장 시스템은 수용된 각 고객들에게 상영률을 보장하도록 데이터를 서비스해야 한다. 저장 시스템은 서비스 사이클마다 시스템 버퍼 관리자에게 다음 사이클에 필요한 블록의 정보를 전달하여 블록을 저장할 가용 버퍼를 요청한다. 시스템 버퍼 관리자는 블록이 캐시된 버퍼가 있다면 그 버퍼를 할당해주고, 없다면 아무 데이터도 저장하고 있지 않은 가용 블록이 있는 지를 조사한다. 있다면 그 블록을 할당해주고, 없다면 블록이 속한 파일의 교체 정책에 따라 버퍼를 선택하여 할당한다.

디스크에서 읽혀 버퍼에 저장된 파일의 블록들은 파일의 참조 유형 속성에 따라 LRU 관리자 또는 DISTANCE 관리자에 의해 관리된다. LRU 관리자는 속성이 반복 참조인 파일들을 관리한다. LRU 관리자는 관리하는 파일들의 버퍼를 최근에 참조된 순서대로 유지하며, 버퍼 교체 시에 가장 최근에 사용되지 않은 버퍼를 선택한다. DISTANCE 관리자는 속성이 순차 참조인 파일들의 버퍼를 DISTANCE 정책으로 관리한다. DISTANCE 알고리즘은 [12]에서 제안한 방법과 같다. DISTANCE 기법에서는 각 고객의 *distance*를 같은 파일을 참조하고 있는 후행 고객간의 데이터 블록 개수로 정의한다. 만일 후행 고객이 없다면 *distance*는 최대값을 가진다. DISTANCE 기법은 고객이 사용하고 반환한 버퍼를 고객 별로 MRU 리스트로서 관리하고, 버퍼를 교체할 때 *distance*가 가장 큰 고객의 버퍼 리스트부터 *distance*가 작은 고객의 버퍼 리스트 순서로 버퍼를 검색하여 버퍼를 선정한다.

(그림 4)는 위에서 설명한 HBM의 수행 알고리즘을 요약해서 보여주고 있다. **HBM_admission_control()**은 저장 시스템에 새로운 고객이 도착했을 때 새 고객의 수용 여부를 검사하기 위해 호출되는 기능이다. **HBM_buffer_allocation()**은 서비스 사이클마다 고객의 버퍼를 할당하는 알고리즘이며, **HBM_buffer_deallocation()**는 서비스 사이클마다 고객이 사용한 버퍼를 반환하는 알고리즘이다.

4. 실 험

4.1 실험 방법

먼저 순차 참조와 반복 참조가 있을 때 기존 버퍼 교체 정책들은 어떠한 성능을 보이는 지를 알아보았다. 다음으로 제안한 혼성 버퍼 교체 기법과 기존 버

```
HBM_admission_control( 입력: 새 고객의 정보, 출력: 수용여부) {
    고객이 참조하려는 파일의 속성을 검사함;
    if(순차 참조) {
        DISTANCE 관리자의 버퍼 공간이 식 (2)를 만족하는 지 검사함;
        if(만족하지 않음)
            return false;
        파일을 DISTANCE 관리자에게 할당함;
    }
    else {
        LRU 관리자의 버퍼 공간이 식 (2)를 만족하는 지 검사함;
        if(만족하지 않음)
            return false;
        파일을 LRU 관리자에게 할당함;
    }
    return true;
}

HBM_buffer_allocation( 입력: 블록의 정보, 출력: 버퍼) {
    블록이 속한 파일의 참조 유형 속성을 검사;
    if(순차 참조) {
        DISTANCE 관리자에게 버퍼를 요청함;
        DISTANCE 관리자는 요청받은 블록을 가진 가용 버퍼가 있는지 검사;
        if(있다)
            버퍼를 할당함;
        else {
            아무런 데이터도 가지지 않은 가용 버퍼가 있는지 검사;
            if(있다)
                그 버퍼를 할당함;
            else
                DISTANCE 정책으로 버퍼를 선택하여 그 버퍼를 할당함;
        }
    }
    else {
        LRU 관리자에게 버퍼를 요청함;
        LRU 관리자는 요청받은 블록을 가진 가용 버퍼가 있는지 검사;
        if(있다)
            버퍼를 할당함;
        else {
            아무런 데이터도 가지지 않은 가용 버퍼가 있는지 검사;
            if(있다)
                그 버퍼를 할당함;
            else
                LRU 정책으로 버퍼를 선택하여 그 버퍼를 할당함;
        }
    }
}

HBM_buffer_deallocation( 입력: 버퍼, 출력: 없음) {
    블록이 속한 파일의 참조 유형 속성을 검사;
    if(순차 참조)
        DISTANCE 관리자에게 버퍼를 반환함;
    else
        LRU 관리자에게 버퍼를 반환함;
}
```

그림 4. HBM의 알고리즘

퍼 교체 기법의 성능 비교를 수행하였다. 실험을 위한 시뮬레이터는 리눅스에서 C 언어로 작성하였다. 측정한 성능 지수는 버퍼 캐시 적중률(hit ratio)이다. 버퍼 캐시 적중률은 버퍼 캐시에서 서비스한 블록 수를 전체 블록 요청 수로 나눈 값이다.

고객은 임의(random)로 도착하며, 고객의 도착 간격(interarrival time)은 지수 분포를 따른다. 고객의 평균 도착 간격은 100초를 사용하였다. 고객이 도착하면 파일을 선택하고 그 파일에 대한 참조 열을 생성한다. 참조 열을 생성할 때 시뮬레이터에 정의된 루프 파라미터, 즉 평균 루프 길이, 평균 루프 횟수, 평균 루프간 간격의 값을 이용하여 참조 열을 만든다. 서비스 라운드마다 각 고객은 파일의 참조 열을 사용하여 파일의 블록을 접근하게 된다. 서비스 라운드는 1초이고 서비스 라운드마다 한 블록을 참조하는 것으로 가정하였다. 파일의 개수는 총 20개이며 모든 파일의 길이는 60분으로 하였다. 파일의 참조 유형은 파일의 인기도와 상관없기 때문에 파일은 임의(random)로 선택하여 참조 유형이 성능에 어떠한 영향을 주는 지를 실험하였다. 각 실험은 8시간동안 수행하였으며 블록 요청 수와 캐시 적중 수를 측정하였다.

표 3 실험에서 사용한 파라미터들의 값

파라미터	값
고객의 평균 도착 간격	100초
파일의 개수	20개
파일의 길이	1시간
서비스 라운드 크기	1초

4.1 기존 버퍼 교체 기법의 성능

본 절에서는 기존 버퍼 교체 기법인 LRU, DISTANCE, MRU 기법의 성능을 살펴본다. 실험 결과를 통해서 새로운 버퍼 교체 기법의 필요성을 알 수 있다.

(그림 5)는 LRU, DISTANCE, MRU를 실험하여 구한 캐시 부재율(miss ratio)이다. 범례에서 L은 LRU, D는 DISTANCE, M은 MRU를 나타내고 SEQ는 순차 참조만 있는 경우이다. 숫자 40-5-200은 파일에 대한 루프 파라미터로서 루프의 평균 길이가 40초, 루프의 평균 반복 횟수 5회, 평균 루프간 간격이 200초임을 의미한다. 이것은 루프가 평균 200초마다 발생하고 루프의 길이가 평균 40초이고 평균 5회 반복되는 것을 뜻한다.
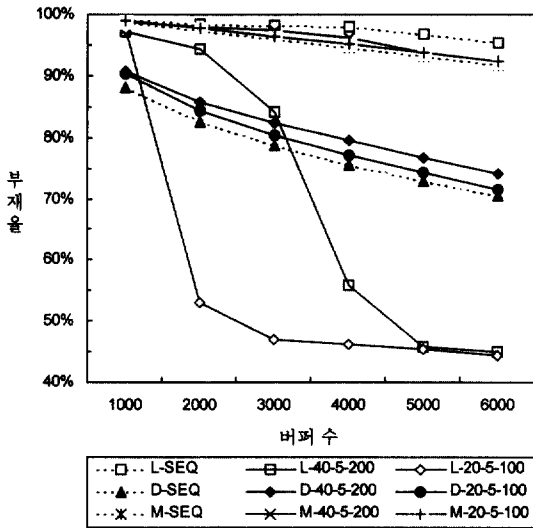
그림 5. 반복 참조가 있는 경우 LRU, DISTANCE, MRU 기법의 성능

이 실험을 통해 다음과 같은 사항을 알 수 있다.

첫째, 순차 참조만 있는 경우 LRU와 MRU 정책의 성능은 매우 좋지 않다. 두 정책의 캐시 부재율이 90 - 99%에 이르고 있다. 그러나, DISTANCE는 캐시 부재율이 70 - 90%이므로 LRU와 MRU보다 우수하다. 예로써 버퍼 수가 6,000일 때 LRU와 MRU의 캐시 부재율은 90% 이상이고 DISTANCE는 71%이므로 약 20% 이상 성능이 좋은 것을 알 수 있다. 따라서, 멀티미디어 파일의 순차 참조에는 DISTANCE 정책이 좋은 캐시 성능을 보인다고 할 수 있다.

둘째, 반복 참조가 있는 경우, MRU와 DISTANCE의 성능은 순차 참조만 있는 경우와 비슷하다. DISTANCE의 경우는 루프 파라미터 유형에 따라 약간의 차이만을 보이고 있어 반복 참조에 큰 영향이 없는 것을 알 수 있다. 그러나, LRU의 경우에는 루프간 간격이 작고 루프 길이가 짧아질수록 성능이 좋아진다. 예를 들어, 버퍼 수가 3,000일 때 루프간 간격이 200이고 루프 길이가 40이면 DISTANCE가 LRU보다 우수하지만, 루프간 간격이 100이고 루프 길이가 20일때는 LRU가 DISTANCE보다 약 34% 더 우수하다. 따라서, 반복 참조가 존재하는 파일의 경우에는 버퍼양이 충분하다면 LRU의 성능이 더 우수하다고 볼 수 있다.

다음으로 평균 루프 길이가 미치는 영향을 알아보았다. 평균 루프간 간격은 100초를 사용하고 루프 횟수는 5회를 사용하였다. 버퍼 수는 4,000, 5,000, 6,000 개이다. (그림 6)은 평균 루프 길이를 20, 40, 60, 80, 100초로 변화를 주어 측정한 캐시 부재율이다. (그림 6)의 범례에서 숫자는 버퍼 수이다.

그림에서 다음과 같은 사항을 관찰할 수 있다.

첫째, LRU의 경우 루프 길이가 길어질수록 캐시 부재율이 증가하고 있다. 이것은 루프 내에서는 순차 참조를 하는데 루프 길이가 길어지면 순차 참조의 효과가 커지게 되기 때문이다. DISTANCE의 경우는 루프 길이에 큰 영향을 받지 않고 있다. 따라서, 루프 길이가 작으면 LRU가 좋고 루프 길이가 크면 DISTANCE가 좋은 경우가 생긴다. 예를 들면, (그림 6)에서 버퍼 수가 6,000개일 때, 루프 길이가 60보다 작으면 LRU가 우수하지만 루프 길이가 60보다 크면 DISTANCE가 우수하다.

둘째, 버퍼 수가 긴 루프를 포함할 정도로 충분하게 많다면 캐시 부재율이 떨어지게 된다. 예를 들면, 그림에서 루프 길이가 40일 때 버퍼 수가 4,000이면 DISTANCE가 LRU보다 우수하지만 버퍼 수가 5,000과 6,000이면 LRU가 더 우수하다. 이것은 버퍼 수가 늘어나면서 루프의 블록들을 버퍼에서 포함하게 되었기 때문이다.

다음으로 루프 횟수가 미치는 영향을 알아보았다. (그림 7)은 루프 횟수를 3, 5, 10회로 변화를 주어 측정한 캐시 부재율이다. 평균 루프 길이는 20초이고 버퍼 수는 4,000개이다. 그림의 범례에서 숫자는 평
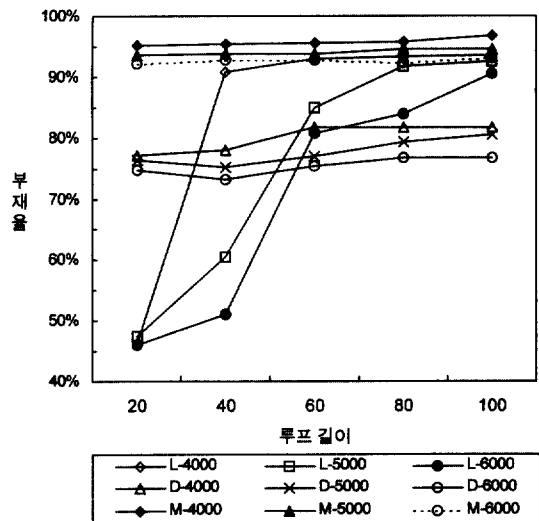
그림 6. 루프 길이에 대한 버퍼 정책들의 성능
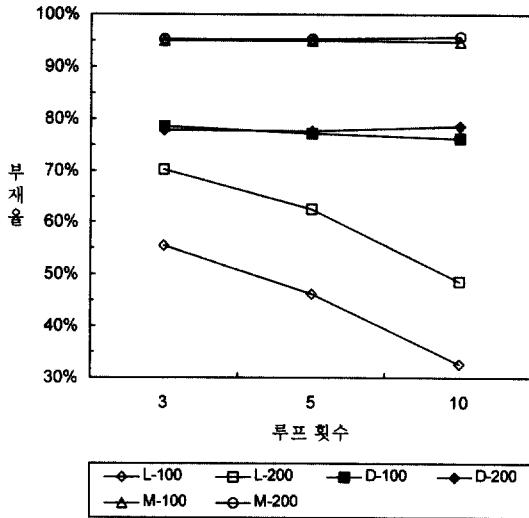
그림 7. 루프 횟수에 대한 버퍼 정책들의 성능

균 루프간 간격이다.

(그림 7)에서도 반복 참조가 있으면 LRU, DISTANCE, MRU 순서로 성능이 좋은 것을 알 수 있다. 또한, 루프 횟수가 많아지면 LRU는 캐시 성능이 더욱 좋아지며, DISTANCE와 MRU는 거의 변화가 없는 것을 알 수 있다.

실험을 통해 멀티미디어 파일에 대한 반복 참조가 있는 경우 LRU가 다른 교체 정책보다 좋은 성능을 보임을 알 수 있었다. 따라서, 순차 참조와 반복 참조가 혼재한다면 참조 유형 별로 적합한 버퍼 교체 정책을 적용하는 새 교체 기법이 필요하다.

4.2 HBM의 성능

제안한 HBM은 참조 유형이 다른 여러 파일이 혼재하는 경우에 그 성능이 우수할 것이다. 제안한 HBM의 성능을 평가하기 위해서 <표 2>와 같은 네 가지의 참조 유형 워크로드를 만들었다. 모든 워크로드에서 평균 루프 길이는 20초, 루프 횟수는 5회를 사용하였다. 시스템의 버퍼 수는 8,000개를 사용하였다. 고객의 파일 선택은 임의로 하였다. 따라서, 모든 파일의 참조 고객 수는 균등하므로 DISTANCE 관리자와 LRU 관리자에 할당하는 버퍼 양은 파일의 속성으로 결정하였다. 즉, 파일의 속성이 반복 참조인 것과 순차 참조인 것의 비율로 각 관리자에 할당되는 버퍼 양을 정했다.

(그림 8)은 워크로드 별로 HBM, LRU, DISTANCE의 성능 결과를 보이고 있다.

표 2. 실험에서 사용한 워크로드

워크로드	설 명
T1	반복 참조 속성의 파일 10개 - 평균 루프간 간격이 100인 파일 10개 순차 참조 속성의 파일 10개
T2	반복 참조 속성의 파일 10개 - 평균 루프간 간격이 100인 파일 5개 - 평균 루프간 간격이 200인 파일 5개 순차 참조 속성의 파일 10개
T3	반복 참조 속성의 파일 15개 - 평균 루프간 간격이 100인 파일 5개 - 평균 루프간 간격이 200인 파일 5개 - 평균 루프간 간격이 500인 파일 5개 순차 참조 속성의 파일 5개
T4	반복 참조 속성의 파일 5개 - 평균 루프간 간격이 100인 파일 5개 순차 참조 속성의 파일 15개

먼저 LRU와 DISTANCE를 비교해보자. T4를 제외하고 LRU가 DISTANCE보다 좋은 성능을 보이고 있다. (그림 5)에서 볼 수 있는 것처럼 평균 루프간 간격이 100이고 평균 루프 길이가 20일 때 LRU는 약 45%의 매우 낮은 버퍼 부재율을 가진다. 반면에 DISTANCE는 70 - 80%의 높은 버퍼 부재율을 가진다. 따라서, T1, T2, T3의 경우 LRU가 DISTANCE보다 좋아지는 것으로 분석된다. 한편, T4에서 LRU가 DISTANCE보다 좋지 않은 것은 파일의 대부분
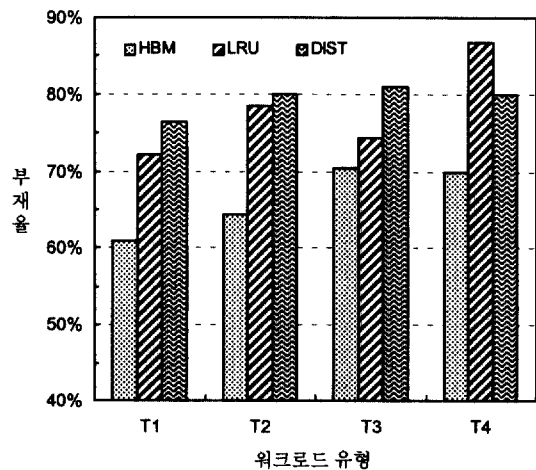
그림 8. HBM의 실험 결과

이(20개 중 15개) 순차 참조인데 LRU는 순차 참조에서 매우 높은 버퍼 부재율(약 95%)을 가지기 때문이다.

HBM은 모든 워크로드 유형에서 가장 좋은 성능을 보이고 있다. 예를 들어, 워크로드 T1에서는 HBM의 캐시 성능이 DISTANCE보다 약 15%가 좋아졌고, LRU보다는 약 11%가 좋아졌다. 이것은 반복 참조 속성의 파일은 LRU로 관리하고, 순차 참조 속성의 파일은 DISTANCE로 관리하여 전체적인 버퍼 캐시 부재율이 다른 기법보다 낮아졌기 때문이다.

5. 결론 및 향후 연구

동영상을 이용한 원격 교육과 같은 여러 멀티미디어 응용 분야가 가능해지면서 멀티미디어 파일 시스템의 데이터 참조 유형이 다양해질 것으로 보인다. 특히, 멀티미디어 교육 시스템의 경우 멀티미디어 파일을 순차적으로 참조할 뿐만 아니라 부분적으로 반복 참조하는 경우가 있을 수 있다. 저장 시스템의 버퍼 캐시의 성능은 응용들의 데이터 참조 유형에 따라 적절한 버퍼 교체 정책을 적용할 때 좋은 성능을 가질 수 있다.

본 논문은 멀티미디어 저장 시스템에서 파일별로 순차 참조와 반복 참조가 혼재하는 경우, 기존의 버퍼 캐시 기법들의 성능을 살펴보고 새로운 버퍼 관리 기법인 혼성 버퍼 관리(HBM) 기법을 제안하였다. 제안한 HBM은 파일별로 참조 유형을 고려하여 적절한 버퍼 정책을 적용한다. 파일마다 참조 유형 속성을 가지고 있으며 이 속성은 응용 수준에서 파일 시스템에게 알려줄 수 있다. 파일 시스템은 파일의 참조 유형 속성을 이용하여 순차 참조 속성이면 DISTANCE 버퍼 교체 정책을 적용하고 반복 참조 속성이면 LRU 버퍼 교체 정책을 적용한다. 실험 결과, HBM은 반복 참조 속성의 파일과 순차 참조 속성의 파일이 혼재하는 경우에 LRU나 DISTANCE보다 더 좋은 성능을 보임을 알 수 있었다.

향후 과제로는 버퍼 정책 관리자 별 버퍼 할당 방법의 개선과 실제 시스템의 워크로드를 구하여 성능을 실험하는 것이 과제로 남아있다. 또한, 파일의 참조 유형 속성을 시스템 수행 중에 동적으로 탐지하여 버퍼 정책을 적용하는 방법도 연구할 계획이다.

참 고 문 헌

[1] D. J. Gemmel, H. M. Vin, D. D. Kandler, P. V. Rangan, and L. A. Rowe, "Multimedia Storage Servers : A Tutorial," IEEE Computer, pp.40-49, May. 1995.

[2] B. Özden, R. Rastogi, and A. Silberschatz, "A Framework for the Storage and Retrieval of Continuous Media Data," Proc. of the IEEE International Conference on Multimedia Computing and Systems, May. 1995.

[3] H. M. Vin and P. V. Rangan, "Designing a Multi-User HDTV Storage Server," IEEE Journal on Selected Areas in Communications, pp. 153-164, Jan. 1993.

[4] Coffman, E. G. and P. J. Denning, Operating Systems Theory, Prentice-Hall, Englewood Cliffs, N. J., 1973.

[5] King, W. F., "Analysis of Paging Algorithms," In Proc. IFIP Congress, pp.485-490, Ljublanjana, Yugoslavia, Aug. 1971.

[6] Lang, T., C. Wood, and I. B. Fernandez, "Database Buffer Paging in Virtual Storage Systems," ACM Transactions on Database Systems, Vol. 2, No. 4, pp.339-351, Dec. 1977.

[7] Rao, G. S., "Performance Analysis of Cache Memories," Journal of the ACM, Vol. 25, No. 3, pp. 378-395, Jul. 1978.

[8] A. Dan and D. Towsley, "An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes," ACM SIGMETRICS, May. 1990.

[9] Y. Smaragdakis, S. Kaplan, and P. Wilson, "EELRU: Simple and Effective Adaptive Page Replacement," ACM SIGMETRICS, 1999.

[10] J. Robinson and M. Devarakonda, "Data Cache Management Using Frequency-Based Replacement," ACM SIGMETRICS, pp.134-142, 1990.

[11] E. J. O'Neil, P. E. O'Neil, and G. Weikum, "The LRU-K Page Replacement Algorithm for Database Disk Buffering," Proc. of the 1993 ACM SIGMOD Conference, pp.297-306, 1993.

[12] B. Özden, R. Rastogi, and A. Silberschatz, "Buffer Replacement Algorithms for Multimedia Storage Systems," Proc of IEEE International Conference on Multimedia Computing and Systems, Jun. 1996.

[13] A. Dan and D. Sitram, "Buffer Management Policy for an On-Demand Video Server," IBM Research Report, RC 19347, Yorktown Heights, NY. 1993.

[14] A. Dan and D. Sitram, "A Generalized Interval Caching Policy for Mixed Interactive and Long Video Environments," IS&T SPIE Multimedia Computing and Networking Conference, Jan. 1996.

[15] A. Reddy and J. Wyllie, "Disk Scheduling in a Multimedia I/O System," Proc. of ACM Multimedia, pp.225-233, 1993.

[16] K. Wu and P. S. Yu, "Consumption-Based Buffer Management for Maximizing System Throughput of a Multimedia System," Proc. of IEEE International Conference on Multimedia Computing and Systems, Jun. 1996.

[17] F. Tobagi, J. Pang, R. Baird, and M. Gang, "Streaming RAID-A Disk Array Management System for Video Files," Proc. Of ACM Multimedia, pp.383-400, Aug. 1993.

[18] M. Chen, D. Kandler, and P. S. Yu, "Optimization of the Grouped Sweeping Scheduling (gss) with Hetetogeneous Multimedia Streams," ACM Multimedia'93, pp.235-242, 1993.

[19] S. Kang and H. Yeom, "Statistical Admission Control for Soft Real-Time VOD Servers," ACM SAC'2000, pp.579-584, 2000.

[20] E. Chang and H. Garcia-Molina, "Effective Memory Use in a Media Server," Proc. of the 23rd VLDB Conference, pp.496-505, Aug. 1997.

[21] Y. S. Ryu and K. Koh, "A Dynamic Buffer Management technique for Minimizing the Necessary Buffer Space in a Continuous Media Server," Proc. of the IEEE International Conference on Multimedia Computing and Systems, Jun. 1996.

[22] J. Nang and S. Heo, "An Efficient Buffer Management Scheme for Multimedia File System," IEICE Transaction of Information and Systems, Vol. E83-D, No. 6, Jun. 2000.

[23] P. Cao, E. W. Felten, and K. Li, "Implementation and Performance of Application Controlled File Caching," Proc. of the 1st USENIX Symposium on Operating Systems Design and Implementation, pp. 165-178, 1994.

[24] J. Choi, S. Noh, S. Min and Y. Cho, "An Implementation Study of a Detection-based Adaptive Block Replacement Scheme," 1999 USENIX Annual Technical Conference, pp.239-252, ACM, 1999

[25] R. H. Patterson, G. A. Gibson, E. Ginting, D. Stodolsky, and J. Zelenka, "Informed Prefetching and Caching," Proc. of the 15th Symposium on Operating System Principles, pp. 1-16, 1995.

류 연 승

1990년 서울대학교 계산통계학과 학사
1992년 서울대학교 전산과학과 석사
1996년 8월 서울대학교 전산과학과 박사
1996년 9월~2000년 8월 삼성전자 연구원
2000년 9월~현재 한림대학교 정보통신공학부 전임강사
관심분야 : 멀티미디어 운영체제, 멀티미디어 네트워크